RANDOM ACCESS MEMORY INITIALIZATION

Field of the invention

The present invention relates to the initialization of Random Access Memory (RAM).

Background

In packet-based communication systems, an initiator sends a read-request packet to a target device. The target sends back a response data packet after some time. For the effective transmission of packets, the device should be able to send many read-request packets before receiving response packets from the target. To support this, the initiator assigns a token/tag to each read-request packet. The initiator can thus identify and handle response packets. The target should use the same token/tag in a response packet. The token is freed up and can be reused once this two-way transaction is completed. For each outstanding token, the initiator needs to remember properties such as address, data length, etc., of the read-request-packet, until a response packet is received.

At the start of a session all tokens are available and can be issued in any order. Once all the tokens are consumed, however, a new read-request packet can be formed only after a response packet is received to release the associated token. This situation occurs also in other applications of storage elements beyond packet-based communication systems.

A communications system will typically have 1024 or more tokens, and against each token, information about the read-request packet is stored (e.g. a 32 bit address, 12 bit data length, etc.). A circuit designer is thus forced to make use of SRAM/register-arrays instead of flip-flops to store this information and to save on chip area.

Fig. 1 shows a memory **10** that stores a set of token information **12**, addressed as 0000 to 1023. The Most Significant Bit (MSB) **14** is used to indicate token status; a logic '1' indicates token is in use and a logic '0' indicates a token is free. Token information is output by the read port **22** to generate a respective token **24**. To generate a new token, a memory controller firstly needs to check the status bit to determine whether the token address is already in use or not.

When memory is implemented in SRAM a problem that arises is the initial (power on) values of the SRAM locations cannot be predicted. A discrete "zeroing task" of the

memory addresses thus required upon power-on, or soft reset, so that token assignment can occur in a sequential manner.

Since only one location can be accessed at a time, in this example, initialization requires 1024 clock cycles. As the size of the array increases, the time needed to initialize the RAM will increase, requiring the other parts of the related system to wait until the initialization completes. This can cause considerable performance degradation.

Fig. 2 shows a further implementation, in which an address generator (binary counter) **30** is used to access all locations of the RAM **10** during initialization. By counting through the set of addresses via the write address port **18**, all status bit locations **14** are initialized with known value of logic "0" via the write port **20**.

Once the address counter reaches address no. 1023, an overflow condition is generated, and an RS flip-flop **32** causes a logical multiplexer **34** to connect a token generator **36** to the write address port **18**.

Summary

A full reset of the status bit of each token address space can be avoided by disabling the status bit check during a first round of token assignment. The addresses can then be overwritten as tokens need to be allocated. Once the full set of addresses is used, the status bit check becomes active.

A memory has a set of address spaces to which token data is written and read. Each address space has a token status bit. A token generator allocates token data to the memory address spaces. Upon a reset occurring, a logic circuit provides logic "0" to the token generator indicating the current token is not in use. New token data is allocated to the address spaces sequentially and the respective status bit is updated or maintained as logic "1". When all address spaces are allocated, the logic circuit provides the actual state of the status bit to the token generator to control subsequent allocations.

This arrangement helps in avoiding the full reset of the status bits in the memory and thereby saving the clock cycles required to initialize them.

Description of drawings

Fig. 1 is a known memory circuit.

Fig. 2 is a known memory arrangement having reset logic.

Fig. 3 is a memory arrangement embodying the invention.

Fig. 4 is a flow diagram relating to the arrangement of **Fig. 3**.

Detailed description

Fig. 3 shows a memory arrangement embodying the invention, and should be read in conjunction with the flow diagram of **Fig. 4**.

On the occurrence of a reset event (step **50**), a logic signal is clocked to the Reset input of the token generator **36** and to the S terminal of the RS flip-flop **32**. The Q output of the flip-flop **32** is thus set to logic 1, indicating an initialisation phase (step **52**). Logic 1 appears at the S terminal of a logical multiplexer **40** causing output of logic 0 (step **54**) at the multiplexer output O. This is also fed-back to the token generator **36** indicating that all the tokens are free to use and can be issued in a sequence (step **56**). The token status is thus assumed to be logic 0, indicating that no tokens are in use eventhough tokens may be existing within the memory address locations.

The token generator **36** is free to issue tokens and update the address spaces **12** via the write address port **18**, typically, in this example, commencing from address 000 and progressing to address 1023. Any existing token is over-written. The respective status bit **14** is updated or remains with logic 1 as each token data is written to an address **12** (step **60**).

The token generator **36** determines whether all addressed spaces have been used (step **62**) and, if so, determines the end of the initialisation phase (step **64**), and logic 1 appears on the overflow output of the token generator **36** to the R input of the flip-flop **32**. This forces the Q output of the flip-flop **32** to logic 0, which, in turn, sets the output of the multiplexer **40** to the value appearing at the "0" input thereof. The "0" input of the

multiplexer **40** receives the most significant bit of the token data **24** that has been pointed to, which, in turn, is provided by the output "O" of the multiplexer **40** to the token generator **36**. In this way, the token generator **36** points to an address and receives an indication of whether a token can be issued and the corresponding memory location can be written based on the respective status bit **14** (step **66**). Token data can now be written to available address spaces as they become available in the usual course of operation of the memory **10** (step **68**).

In the conventional method described with reference to the example of **Fig. 2**, at least 1024 clock cycles are needed to initialize the status bit of the RAM. In the embodiment described, there is no need to wait for the initialization of status bits **14** in RAM **10**. Tokens can be issued immediately after reset. In effect the initialization gets completed in the background when all tokens are issued a first time.

Conclusion

A memory having 1024 address locations is understood to be merely exemplary. Various alterations and modifications can be made to the techniques and arrangements described herein, as would be apparent to one skilled in the relevant art.